Synthesis of Non-Replicated Dynamic Fragment Allocation Algorithm in Distributed Database Systems

Nilarun Mukherjee
Senior Lecturer, Department of CSE/IT
Bengal Institute of Technology, Kolkata, West Bengal, India
nilarun.mukherjee@gmail.com

***Abstract*— Distributed databases have become the most essential technology for recent business organizations. In most of the cases Distributed Databases are developed in a Bottom-Up approach, fragmentation exists beforehand. The optimum allocation of those fragments is the only way, which can be exploited by the designers to increase the performance, efficiency, reliability and availability of the Distributed Database in the realistic dynamic environment, where the access probabilities of nodes to fragments change over time. In this paper, a new dynamic fragment allocation algorithm is proposed in Non-Replicated allocation scenario, which incorporates the time constraints of database accesses, volume threshold and the volume of data transmitted in successive time intervals to dynamically reallocate fragments to sites at runtime in accordance with the changing access patterns. It will migrate a fragment to a particular site only when that site has made data transfer higher than all other sites and greater than the threshold value from or to that fragment consistently in some recent time intervals. The choice of the volume threshold, the number of time intervals and the duration of each time interval are the most important factors which regulate the frequency of fragment reallocations. The proposed algorithm will decrease the movement of fragments over the network and data transfer cost and will also improve the overall performance of the system by dynamically allocating fragments in a most optimum and intuitive manner.**

***Index Terms*— Distributed Database, Fragmentation, Dynamic Fragment Allocation, Distributed Transaction, Local Agent, Root Agent**

I. INTRODUCTION

A Distributed Database is a collection of data, which logically belong to the same system, but are distributed over the sites of a computer network. Each site of the network has autonomous processing capability and can perform local database applications [9]. Each site also participates in the execution of at least one global database application, which requires accessing data residing at several different sites from geographically dispersed locations, using a communication subsystem.

The primal motivations for distributed databases are to improve performance, to increase the availability of data, shareability, expandability and access facility. In a distributed database, maximization of the locality of processing of database applications by allocating data as close as possible to the applications which use them, significantly reduces the communication overhead with respect to a centralized database.

In most of the cases Distributed Databases are developed in a Bottom-Up approach, where several already existing centralized databases located at several geographically dispersed sites are integrated to form the Distributed Database. In this scenario, there is no scope of fresh fragmentation, as the fragments exists beforehand and they are analyzed and integrated to form the global relations. Moreover, fragmentation depends highly on the business and organizational requirements. Therefore, the optimum allocation of those fragments is the only way, which can be exploited by the designers to increase the performance, efficiency, reliability and availability of the Distributed Database. The main aim is to maximize the locality of processing for each distributed application by storing the fragments closer to where they are more frequently used in order to achieve best performance. This means placing data i.e. fragments required by the distributed applications at their site of origin or at sites which are closer to their site of origin. I.e. to maximize the "local" references and to minimize "remote" references to the data by each distributed application with respect to its site of origin [9]. This is done by adding the number of local and remote references corresponding to each candidate fragmentation and fragment allocation, and selecting the best solution among them, i.e. the solution providing highest processing locality or *Complete Locality* to maximum number of distributed applications. The Complete Locality means the distributed application can be completely executed at its site of origin; thus reducing overall remote accesses and simplifying the control mechanism for the execution of the distributed applications. A poorly designed data allocation can lead to inefficient computation, high access cost and high network loads.

Various approaches have already been evolved for dynamic allocation of data in distributed database [1] to improve the database performance. These address the more realistic dynamic environment, where the access probabilities of nodes to fragments change over time. Fragment allocation can further be divided in two ways:

A.. Non – Redundant Fragment Allocation:

Each fragment of each global relation is allocated to exactly one site.

DOI: 01.IJIT.01.01.98

B. Redundant Fragment Allocation:

Fragments of each global relation are allocated to one or more sites introducing replication of the fragments. In this paper, a new dynamic fragment allocation algorithm is proposed in Non-Replicated allocation scenario, which incorporates the time constraints of database accesses, the volume threshold and most importantly the volume of data transmitted in successive time intervals to dynamically reallocate fragments to sites at runtime in accordance with the changing access patterns i.e. in accordance with the changing access probabilities of nodes to fragments over time. The proposed algorithm will decrease the movement of fragments over the network and data transfer cost and will also improve the overall performance of the system by dynamically allocating fragments in a most optimum and intuitive manner.

II. RELATED WORK

Till date many works have been published on the problem of allocation of data or fragments of the global relations to geographically dispersed sites of a distributed database. [10] considered the problem of file allocation for typical distributed database applications with a simple model of transaction execution. [11] incorporated issues like concurrency and queuing costs. [12] provides an integrated approach for fragmentation and allocation. [12] identified seven criteria that a system designer can use to determine the fragmentation, replication and allocation. [15] presents a replication algorithm that adaptively adjusts to changes in read-write patterns. [16] provides an approach based on Lagrangian relaxation and [17] describes heuristic approaches. More recently [18] has given a high-performance computing method for data allocation in distributed database system.

In most of the above approaches, data allocation has been proposed prior to the design of the distributed database, depending on some static data access patterns and/or query patterns. Static allocation of fragments provides the best solution when the access probabilities of nodes to fragments never change over time, but, degrades performance in a dynamic environment, where probabilities change over time.

Over past few years, work has been introduced for dynamic fragment allocation in distributed database systems. [15] gives a model for dynamic data allocation for data redistribution and incorporates a concurrency mechanism. In [4] an algorithm is proposed for dynamic data allocation, which reallocates data with respect to the changing data access patterns. [13] provides approach for allocating fragments by adapting a machine learning approach. [7] considers incremental allocation and reallocation based on changes in workload. [19] incorporated security considerations into the dynamic file allocation process. In [20] an optimal fragment allocation algorithm for non-replicated distributed database systems is proposed. [21] has introduced a threshold algorithm for non-replicated fragment allocation in distributed databases. In the threshold algorithm, the fragments are continuously reallocated according to the changing data access patterns.

In this paper, a new dynamic fragment allocation algorithm is proposed in Non-Replicated allocation scenario. The proposed algorithm incorporates the volume threshold, the time constraints of database accesses and most importantly the volume of data transmitted in successive time intervals to dynamically reallocate fragments to sites at runtime in accordance with the changing access patterns i.e. in accordance with the changing access probabilities of nodes to fragments over time. It will migrate a fragment to a particular site only when that site has made data transfer higher than all other sites and also greater than the threshold value from or to that fragment consistently over some recent time intervals. The choice of the volume threshold, the number of time intervals and the duration of each time interval are the most important factors which regulate the frequency of fragment reallocations i.e. the performance of the algorithm.

III. DESIGN

The main factor that affects the efficiency and turnaround time of the applications of a distributed database is the time delay for transferring the data needed by a certain distributed database query or application, over the network from the fragments located at several different geographically dispersed sites to the site of origin of the application or query. The primary aim is to allocated fragments needed by certain distributed database application either at the same site where the distributed database application / query is invoked or at sites closer to the site of origin, so that the data transmission over the network is minimized during the execution of the application / query. It is a considerably complex problem, especially, in the scenario where the probability or pattern of accessing fragments by the distributed database applications at different sites changes dynamically over time. Also, it is not a feasible solution to place the entire data of a distributed database at every site of the system; it will violate the basic requirements of a distributed database [9].

The proposed algorithm for dynamic fragment allocation in distributed database exploits the concepts of the existing algorithms: the Optimal Algorithm [4], the Threshold Algorithm [21] and TTCA Algorithm [1]. In distributed database the efficiency and turnaround time of the distributed database applications ultimately depends on the volume of data that is required to get transferred over the network from one site to another due to the execution of those applications. Reallocating a fragment from a certain site to another site, which makes the highest number of accesses to that fragment and not to the site, which even though does not make the highest number of accesses to that fragment but results in transmission of maximum volume of data from or to that fragment, will not be always beneficial i.e. will not always improve the overall performance or efficiency of the distributed database.

The main agenda of the proposed algorithm is to reallocate a fragment located at a certain site to another site,

DOI: 01.IJIT.01.01.98

which has made highest data transmission from or to that fragment consistently over time, not to the site making highest or frequent accesses but transferring comparably small amount of data from or to that fragment. The value of the Consistency Threshold (ë) determines the number recent time intervals for which a particular site has to consistently make data transmission higher than all other sites and also greater than the data Volume Threshold (í) from or to a particular fragment located at another site, to get that fragment reallocated at this new site. The duration of each time interval is determined by the Time Constraint for Fragment Reallocation (ô).

Moreover, most of the time the site which consistently makes the highest number of accesses to a particular fragment located at a particular site in some recent time intervals also results in transmission of maximum volume of data from or to that fragment in that time duration. Thus, for most of the cases the proposed algorithm for dynamic fragment allocation will reflect the most realistic scenario and will result in more intuitive and optimum dynamic fragment reallocation, in Non-Replicated allocation scenario. Furthermore, reallocation of fragments from one site to another site over the network just in order to make the distributed database applications to have their required data at their site of origin or at the sites very closer to their site of origin, incurs huge data transmission over the network and results in transmission overload. Because, fragments are expected to contain huge amount of data as compared to the data required by and transmitted due to the execution of those distributed database applications. Thus, it is not desirable to have frequent migrations of fragments over the network, as this can significantly affect or degrade the overall performance or efficiency of the distributed database.

The proposed dynamic fragment allocation algorithm migrates a fragment located at a certain site to another site, which has made highest data transmission from or to that fragment than all other sites and also greater than the Volume Threshold for Fragment Reallocation (í) consistently over at least ë + 1 recent time intervals, where ë is the Consistency Threshold and the duration of each time interval is determined by the Time Constraint for Fragment Reallocation (ô). *The Algorithm:*

1. Initially all the fragments of all the global relations of the distributed database are distributed over different sites using any static allocation method in non-replicated manner. Let there be total N number of fragments of global relations distributed among the total M number of sites in the distributed database system. Each site has one or more fragments allocated to it. A distributed database query or application may require accessing several different fragments allocated at several different sites for execution.
2. Each site maintains a separate data structure named *Access Log*, which stores certain information regarding each access to the fragments allocated at that site, by the distributed database queries or applications invoked at the same or different sites. Each *Access Log* record denoted by A_k^h (i.e. k^{th} access at site h, where k = 1,2,3,…to infinity and h = 1,2,3,...M) stores the following information regarding each access:
   - Name or Identifier of the Fragment accessed.
   - Address of the accessing site.
   - Date and Time of the access.
   - The volume of data transmitted from or to that fragment in Bytes.
3. Three parameters are used to tune the performance of the algorithm:
   - Time Constraint for Fragment Reallocation (ô).
   - Volume Threshold for Fragment Reallocation (í).
   - Consistency Threshold (ë).
4. The proposed algorithm needs each site to maintain a separate data structure named Access Counter for each fragment located at that site, which stores the following information after each access to that fragment:
   - Candidate Site Address: The address of the site which has made data transmission from or to that fragment higher than all other sites and also greater than í in the current time interval. Initially it is set to the address of the site where the fragment is located.
   - Number of Recent Time Intervals: It provides the number of recent time intervals for which the Candidate Site has made data transmission from or to that fragment higher than all other sites and also greater than í . Initially it is set to the value of the Consistency Threshold (ë) + 1.

The following Steps [5 to 7] are performed at each site for each individual access to a fragment allocated at that site by a certain distributed database query or application invoked at the same or different site. Suppose at site h an access is made and processed for fragment i allocated at that site from site j at time t, where h = 1,2,3,…M, i = 1,2,3,...N and j = 1,2,3,...M and h = j or h ‘“ j. The local agent at site h performs the following operations:

5. Write a log record A_k^h in the Access Log at site h.
6. Calculate the total volume of data transmitted (in bytes) in between the fragment i and all the sites (including site h) where from the accesses to the fragment i located at site h are made, through the accesses occurred within the time interval ô up to current access time t. $A_k^h V_i^m$ denotes the volume of data transmitted (in bytes) in between the fragment i allocated at site h and the site m in the access A_k^h at certain point of time, where m = 1,2,3,...M. The total volume of data transmitted (denoted by $V_i^m t$) in between the fragment i allocated at site h and the site m through the accesses occurred within the time interval ô up to current access time t is calculated as:

   $V_i^m t =$ “ $A_k^h V_i^m$

   Where A_k^h occurred within the time interval ô up to current access time t. $V_i^m t$ is calculated for all the sites (including site h) where from the accesses to the fragment h are made, through the accesses occurred within the time interval ô up to current access time t.
7. If the total volume of data transmitted (in bytes) in between the fragment i allocated at site h and the site j through the

DOI: 01.IJIT.01.01.98

accesses occurred within the time interval ô up to current access time t is greater than í and is also greater than the total volume of data transmitted (in bytes) in between the fragment i and all other sites (including site h) where from the accesses to the fragment i located at site h are made, through the accesses occurred within the time interval ô up to current access time t, i.e. if and only if $V_i^j t > í$ and $V_i^j t > V_i^m t$ and where j, m = 1,2,3,...M and m '“ j, then in the Access Counter for fragment i at site h the following operations are executed, otherwise do nothing:

- If the address of the accessing site in the log record A_k^h is same as the address of site h, i.e. the access is made from the same site (h = j), then:
  - i. If the Candidate Site Address in the Access Counter for fragment i is same as the address of site h, then do nothing as fragment i is already allocated to site h.
  - ii. Else if the Candidate Site Address in the Access Counter for fragment i is not same as the address of site h, then update the Candidate Site Address in the Access Counter for fragment i with the address of site h and set the value of the corresponding Number of Recent Time Intervals to Consistency Threshold (ë) + 1.
- Else if the address of the accessing site in the log record A_k^h is different from the address of site h, i.e. the access is made from a different site (h '“ j); then:
  - iii. If the Candidate Site Address in the Access Counter for fragment i is same as the address of site j, then increment the corresponding Number of Recent Time Intervals by 1. The incremented value indicates that site j has made data transmission from or to that fragment higher than all other sites and also greater than í from or to that fragment for that number of recent time intervals.
  - iv. If the updated value of the Number of Recent Time Intervals in the Access Counter for fragment i is greater than ë, then the fragment i is migrated and reallocated to site j and removed from the current site h, catalogs are updated accordingly.
  - v. If the Candidate Site Address in the Access Counter for fragment i is not same as the address of site j, then update the Candidate Site Address in the Access Counter for fragment i with the address of site j and set the value of the corresponding Number of Recent Time Intervals to 1. This indicates that site j has made data transmission from or to that fragment higher than all other sites and also greater than í during the current time interval not in the immediately preceding time interval.

In distributed database, at each site the local agent of a remote or local distributed transaction accesses or interacts with the local database where the fragments of the global relations are actually stored [9]. In case of insertion or modification operations on a fragment, the local agent receives the data to be inserted or to be used to modify the existing data in the particular fragment from the root agent of the distributed transaction. Thereafter, the local agent sends those data along with the command (insert or update) to the local transaction manager, which actually accesses the database table corresponding to that fragment stored at the local database to perform insertion or modification. In case of retrieval, the local agent receives the data retrieved from the database table stored at the local database corresponding to a particular fragment through the local transaction manager. Thereafter, it sends those retrieved data to the root agent of the distributed transaction executing at the same or different site in the distributed database network. In all the cases the local agent of the distributed database transaction temporarily retains the data to be inserted or to be used to modify the existing data in the particular fragment or being retrieved from the particular fragment during the execution of a of the distributed database application or query. Moreover, the local agent belongs to the distributed database transaction management system and it is designed and programmed by the developer of the distributed database management system. Therefore, it is possible to enhance the distributed database transaction management system and the local agents of the distributed database transactions to add the functionality of writing Access Log Records and to calculate the volume of data transmitted from or to each fragment (in Bytes) at each fragment access and to maintain the Access Counter, i.e. to implement the proposed dynamic fragment allocation algorithm.

IV. PERFORMANCE EVALUATION

In Optimal Algorithm [4], initially all the fragments are distributed over the different sites using any static data allocation method. After the initial allocation, system maintains an access counter matrix for each locally stored fragment at each site or node. Every time an access request is made for the locally stored fragment, the access counter of the accessing site for that fragment is increases by one. If the counter of a remote node becomes greater than the counter of the current owner, then the fragment is moved to the accessing node. The problem of Optimal Algorithm [4] technique is that if the changing frequency of access pattern for each fragment is high, then it will spend more time for transferring fragments to different sites and will incur huge data transmission overload. Threshold algorithm [21] decreases the migration of fragments and guarantees the stay of the fragment for at least (ç +1) accesses at the new node after a migration, where ç is the value of threshold. But threshold algorithm resets the counter of local fragment to zero, every time a node is going for a local access and it does not specify which node will be the fragment's new owner when the counter exceeds the threshold value, also it does not give the information about past accesses of the fragments and does not consider the time variable of the access pattern.

The TTCA [1] algorithm removes all the above problems of threshold algorithm [21] by adding a time constraint to consider the time of the accesses made to a particular fragment. TTCA [1] decreases the migration of fragments

DOI: 01.IJIT.01.01.98

over different sites, thus reduces the transmission overload due to fragment migrations as compare to simple Threshold algorithm [21]. TTCA [1] maintains a counter matrix at each site to store the number of accesses made by the sites to a particular fragment located at that site, which is incremented on each access. It reallocates data with respect to the changing data access patterns with time constraint. It migrates a fragment from one site to another site, which has most recently accessed that fragment for ç + 1 numbers of times in a time period ô up to the current access time, where ç is the Threshold Value and ô is the Time Constraint. But TTCA [1] does not store the time of those accesses in that counter matrix. Therefore, it does not provide any conceivable method to determine within which time period the certain number of accesses (determined by the value of the counter) made by a certain site to a particular fragment located at another site have occurred.

The proposed algorithm will remove all the above problems of Threshold [21] and TTCA [1] algorithm. It will dynamically reallocate fragments at runtime in accordance with the changing access probabilities of sites to fragments over time. This dynamic fragment allocation algorithm migrates a fragment located at a certain site to another site, which results in transmission of data higher than all other sites and also greater than the predefined threshold value from or to that fragment consistently over some recent time intervals. It reduces the transfer of data over the network during the execution of the distributed database application / query, in Non-Replicated allocation scenario.

The proposed algorithm improves the overall performance or efficiency of the distributed database by imposing a more strict condition for fragment reallocation in distributed database and resulting in fewer migrations of fragments from one site to other over the network and further reduces the data transmission overload due to fragment migrations as compared to the Optimal Algorithm [4], the Threshold Algorithm [21] and TTCA Algorithm [1]. The choice of the most appropriate values of í, ô and ë serves as the key factors for determining the optimum performance of the algorithm, to minimize unnecessary fragment migrations and data transfer over the network required during the execution of the distributed database queries or applications by increasing their locality of processing i.e. to maximize the overall throughput, performance and efficiency of the distributed database applications. For fixed values of í and ô, if ë is increased, then the migration of fragments from one site to other will sharply decrease vice versa, as it will be very hard for any site to be the highest data transferee from or to a particular fragment consistently over a large number of recent time intervals. I.e.

$$\text{Fragments Migration / Reallocation Frequency} \quad (\psi) \propto \frac{\tau}{\lambda \times \nu}$$

For fixed values of ô and ë, if the value of í is increased, then the migration of fragments from one site to other will decrease vice versa. For fixed values of í and the ë, if the value of ô is increased, then the migration of fragments from one site to other will increase vice versa. The results of several experiments simulating the implementation of the proposed algorithm in the distributed database suggest that the performance of the proposed algorithm i.e. the impact of the proposed algorithm on the performance and efficiency of the distributed database varies widely with different values of the above regulating parameters. Special care and critical analysis should be performed to choose the most appropriate values of the Volume Threshold (í), Time Constraint (ô) and the Consistency Threshold (ë) to highly optimize the algorithm to maximize the performance and efficiency of the distributed database.

If í is made very low i.e. below the average data transfer volume between the sites of the distributed database network, then the proposed algorithm will practically migrate fragments to other sites, depending only on one condition i.e. the site which results in higher transmission of data than all other sites from or to that fragment in some recent time intervals will get that fragment reallocated or migrated to it.

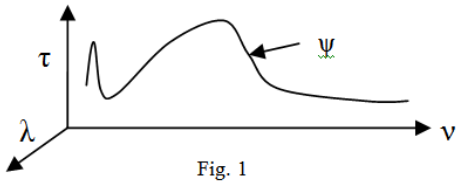

Fig. 1

In Figure 1, the frequency of fragment reallocation (ø) is plotted against í, ë and ô. The impact of í, ë and ô on the rate of fragment migration in the distributed database i.e. on the performance of the proposed dynamic fragment allocation algorithm is clearly evident from the above graphical representation.

If í and ë is decreased and ô is increased, then there will be frequent fragment migrations. On the other hand, if í and ë is increased and ô is decreased, then after a certain point there will rarely be any fragment migrations. Because, it will be extremely hard for any site to be the highest data transferee (higher than all other sites and also greater than í) from or to a particular fragment consistently over a large number of recent short time intervals i.e. to fulfill the condition for fragment reallocation and to get that fragment reallocated to it.

The proposed algorithm requires more storage space and increases the storage cost due to the maintenance of the Access Counter as well as the *Access Log* at the sites in which log records are created for each access. It incurs more computational overhead per database access at each site due to the calculation of the total volume of data transmitted (in bytes) in between that fragment and all the sites where from the accesses to that fragment are made, through the accesses occurred within the time interval ô up to current access time.

But these drawbacks are well compensated by the advantage gained from the significant reduction in the unnecessary fragment migrations as compared to the Optimal Algorithm [4], the Threshold Algorithm [21] and TTCA Algorithm [1], which incurs huge data transmission overload and degrades the overall performance and efficiency of the distributed database. Thus, for most of the cases the proposed algorithm for dynamic fragment allocation will reflect the most realistic scenario and will result in more intuitive and

DOI: 01.IJIT.01.01.98

optimum dynamic fragment reallocation, in Non-Replicated allocation scenario.

V. CONCLUSION

In this paper a new sophisticated dynamic fragment allocation algorithm is proposed in Non-Replicated allocation scenario. The proposed algorithm incorporates the time constraints of database accesses, volume threshold and most importantly the volume of data transmitted in successive time intervals to dynamically reallocate fragments to sites at runtime in accordance with the changing access probabilities of nodes to fragments over time. The proposed algorithm migrates a fragment located at a certain site to another site, which has made higher data transmission from or to that fragment than all other sites and also greater than the Volume Threshold for Fragment Reallocation (í) consistently over at least ë + 1 recent time intervals, where ë is the Consistency Threshold and the duration of each time interval is determined by the Time Constraint for Fragment Reallocation (ô). The proposed algorithm improves the overall performance or efficiency of the distributed database by imposing a more strict condition for fragment reallocation in distributed database and results in significant reduction in the frequency of unnecessary fragment migrations from one site to other over the network and thus substantially reduces the data transmission overload due to fragment migrations as compared to the Optimal Algorithm [4], the Threshold Algorithm [21] and TTCA Algorithm [1]. The results of several experiments simulating the implementation of the proposed algorithm in the distributed database suggest that the choice of the most appropriate values of í, ô and ë serves as the key factors for determining the optimum performance of the algorithm, to minimize the frequency of unnecessary fragment migrations and to minimize the transfer of data over the network required during the execution of the distributed database queries or applications by increasing their locality of processing i.e. to maximize the overall throughput, performance and efficiency of the distributed database applications.

REFERENCES

[1] Arjan Singh and K.S. Kahlon, "Non-replicated Dynamic Data Allocation in Distributed Database Systems", *IJCSNS International Journal of Computer Science and Network Security*, VOL.9 No.9, pp. 176-180, September 2009.

[2] Svetlana Vasileva, Petar Milev, Borislav Stoyanov, "Some Models of a Distributed Database Management System with data Replication", *International Conference on Computer Systems and Technologies-CompSysTech'07*, Vol.–II, pp. II.12.1–II.12.6, 2007.

[3] S. Agrawal, V. Narasayya, and B. Yang, "Integrating Vertical and Horizontal Partitioning into Automated Physical Database Design", *Proc. 2004, ACM SIGMOD International Conf. Management of Data*, pp. 359-370, 2004.

[4] A. Brunstroml, S.T. Leutenegger and R. Simhal, "Experimental Evaluation of Dynamic Data Allocation Strategies in a Distributed Database with changing Workload", *ACM Trans. Database Systems*, 1995.

[5] S. March and S. Rho, "Allocating Data and Operations to Nodes in Distributed Database Design," *IEEE Trans. Knowledge & Data Eng.*, Vol. 7, no. 2, pp. 305-317, 1995.

[6] T. Ulus and M. Uysal, "A Threshold Based Dynamic Data Allocation Algorithm - A Markove Chain Model Approach", *Journal of Applied Science*, Vol. 7(2), pp 165-174, 2007.

[7] A. Chin, "Incremental Data Allocation and Reallocation in Distributed Database Systems", *Journal of Database Management*, Vol. 12, No. 1, pp. 35-45, 2001.

[8] Hideo Hanamura, Isao Kaji, Kinji Mori, "Autonomous Consistency Technique in Distributed Database with Heterogeneous Requirements", *IPDPS Workshops 2000*, pp. 706-712, 2000.

[9] S. Ceri and G. Pelegatti, "*Distributed Database Principles & Systems*", McGraw-Hill International Editions.

[10] S. Ceri, G. Martella, and G. Pelagatti, "Optimal file Allocation for a Distributed Database on a Network of Minicomputers", *in Proc. International Conf. on Database, Aberdeen*, July 1980, British Computer Society Hayden.

[11] S. Ram and S. Narasimhan, "Database Allocation in a Distributed Environment: Incorporating a Concurrency Control Mechanism and Queuing Costs", *Management Science*, vol. 40, no. 8, pp. 969-983, 1994.

[12] K. Karlaplem and N. Pun, "Query-Driven Data Allocation Algorithms for Distributed Database Systems", *Proc. Eighth International Conf. Database and Expert Systems Applications (DEXA '97)*, pp. 347- 356, Sept. 1997.

[13] A. Tamhankar and S. Ram, "Database Fragmentation and Allocation: An Integrated Methodology and Case Study", *IEEE Trans. Systems, Man and Cybernetics—Part A*, vol. 28, no. 3, May 1998.

[14] A. Chaturvedi, A. Choubey, and J. Roan, "Scheduling the Allocation of Data Fragments in a Distributed Database Environment: A Machine Learning Approach", *IEEE Trans. Eng. Management*, vol. 41, no. 2, pp. 194-207, 1994.

[15] O. Wolfson, S. Jajodia, and Y. Huang, "An Adaptive Data Replication Algorithm", *ACM Trans. Database Systems*, vol. 22, no. 2, pp. 255-314, 1997.

[16] G. Chiu and C. Raghavendra, "A Model for Optimal Database Allocation in Distributed Computing Systems", *Proc. IEEE INFOCOM 1990*, vol. 3, pp. 827-833, June 1990.

[17] Y. Huang and J. Chen, "Fragment Allocation in Distributed Database Design", *Journal of Information Science and Eng.*, vol. 17, pp. 491- 506, 2001.

[18] I.O. Hababeh, M. Ramachandran and N. Bowring, "A high-performance computing method for data allocation in distributed database systems", *Springer, Journal of Supercompute*r (2007) 39:3-18.

[19] A. Mei, L. Mancini, and S. Jajodia, "Secure Dynamic Fragment and Replica Allocation in Large-Scale Distributed File Systems", *IEEE Trans. Parallel and Distributed Systems*, vol. 14, no. 9, pp. 885-896, Sept. 2003.

[20] L.S. John, "A Generic Algorithm for Fragment Allocation in Distributed Database System", *ACM* 1994.

[21] T. Ulus and M. Uysal, "Heuristic Approach to Dynamic Data Allocation in Distributed Database Systems", *Pakistan Journal of Information and Technology*, 2(3): pp. 231-239, 2003.

DOI: 01.IJIT.01.01.98